VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

BEZDRÁTOVÝ SBĚR DAT Z BOSCH XDK V LABVIEW
WIRELESS DATA ACQUISITION FROM BOSCH XDK IN LABVIEW

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Martin Hlaváč**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Stanislav Pikula, Ph.D.**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Automatizační a měřicí technika**

Ústav automatizace a měřicí techniky

Student: Martin Hlaváč ***ID:*** 203561
Ročník: 3 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Bezdrátový sběr dat z Bosch XDK v LabVIEW

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je vytvoření aplikace v LabVIEW pro bezdrátový záznam dat ze zařízení Bosch XDK. Aplikace bude umožňovat záznam do různých datových formátů. Bude zvoleno rozhraní Bluetooth nebo Wifi. Bude připraveno několik variant nastavení Bosch XDK pro posílání dat z různých snímačů přes bezdrátové rozhraní. Zadání bakalářské práce lze shrnout do následujících bodů:

1. Popište zařízení Bosch XDK včetně parametrů jeho sensorů, dostupných komunikačních rozhraní a prostředí XDK-Workbench pro programování a konfiguraci zařízení.
2. Proveďte návrh aplikace v LabVIEW pro záznam dat přijímaných prostřednictvím bezdrátových rozhraní.
3. Navrhněte několik scénářů pro posílání dat přes bezdrátová rozhraní a zdokumentujte potřebná nastavení Bosch XDK.
4. Realizujte navrženou aplikaci v LabVIEW, zdokumentujte ji a otestujte funkčnost pro navržené scénáře.

DOPORUČENÁ LITERATURA:

[1] VLACH, Jaroslav, Josef HAVLÍČEK a Martin VLACH. Začínáme s LabVIEW. Ilustroval Viktorie VLACHOVÁ. Praha: BEN - technická literatura, 2008. ISBN 978-8073002459.

[2] BLUME, Peter A. The LabVIEW style book. Upper Saddle River: Prentice Hall, 2007. ISBN 978-0131458352.

Termín zadání: 8.2.2021 ***Termín odevzdání:*** 24.5.2021

Vedoucí práce: Ing. Stanislav Pikula, Ph.D.

doc. Ing. Václav Jirsík, CSc.
předseda rady studijního programu

Abstrakt

Práce pojednává o tom, co je to XDK 110, jaké jsou jeho snímače, jejich vlastnosti a jakými komunikačními rozhraními lze se zařízením komunikovat. Jelikož XDK 110 umožňuje komunikaci pomocí Bluetooth Low Energy (BLE), jsou v práci uvedeny možnosti řešení komunikace pomocí tohoto rozhraní, které zatím není v LabVIEW přímo podporováno. Dále jsou v práci navrženy scénáře pro vyčítání dat z XDK 110 a následně je vypracována a popsána aplikace pro vyčítání a zápis změřených dat z XDK v LabVIEW.

Klíčová slova

LabVIEW, Bosch XDK 110, Bluetooth Low Energy (BLE), USB dongle BLED112-V1

Abstract

The work summarizes what the XDK 110 is, what sensors it has, their properties and what communication interfaces can be used to communicate with the XDK device. Since the XDK 110 enables communication using Bluetooth Low Energy (BLE), this work presents the possibilities of solving communication using this interface, which is not yet directly supported in LabVIEW. Furthermore, scenarios for reading data from XDK 110 are proposed in the work, and then an application for reading and writing measured data from XDK in LabVIEW is developed and described.

Keywords

LabVIEW, Bosch XDK 110, Bluetooth Low Energy (BLE), USB dongle BLED112-V1

Bibliografická citace

HLAVÁČ, Martin. *Bezdrátový sběr dat z Bosch XDK v LabVIEW*. Brno, 2021. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/134833. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce Stanislav Pikula.

Prohlášení autora o původnosti díla

Jméno a příjmení studenta:	*Martin Hlaváč*
VUT ID studenta:	*203 561*
Typ práce:	*Bakalářská práce*
Akademický rok:	*2020/21*
Téma závěrečné práce:	*Bezdrátový sběr dat z Bosch XDK v LabVIEW*

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucího závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

.

V Brně dne: 24. května 2021 ----------

podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Stanislavu Pikulovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne: 24. května 2021 ----------

podpis autora

Obsah

SEZNAM OBRÁZKŮ

Úvod

Tato bakalářská práce se zabývá komunikací XDK 110 a programu LabVIEW. Při práci se všemi snímači je většinou potřebné následné zpracování dat, to můžeme provádět například pomocí LabVIEW. Avšak před zpracováním je prvně potřeba data do LabVIEW dostat a jedním ze způsobů, kterým to jde, je Bluetooth.

Práce se skládá ze čtyř kapitol. V první kapitole práce je popsáno, co je XDK 110, jak ho naprogramovat a jaké má toto zařízení integrované senzory. Dále jsou uvedeny možné prostředky, umožňující komunikaci s LabVIEW. Z těchto prostředků jsou v práci více rozvinuty rozdíly mezi Wi-Fi a Bluetooth.

V další kapitole se práce zabývá LabVIEW a komunikací přes Bluetooth. Jelikož XDK obsahuje vyšší standart Bluetooth nazvaný BLE, který však LabVIEW nepodporuje, jsou popsány i možnosti, jak tento problém řešit.

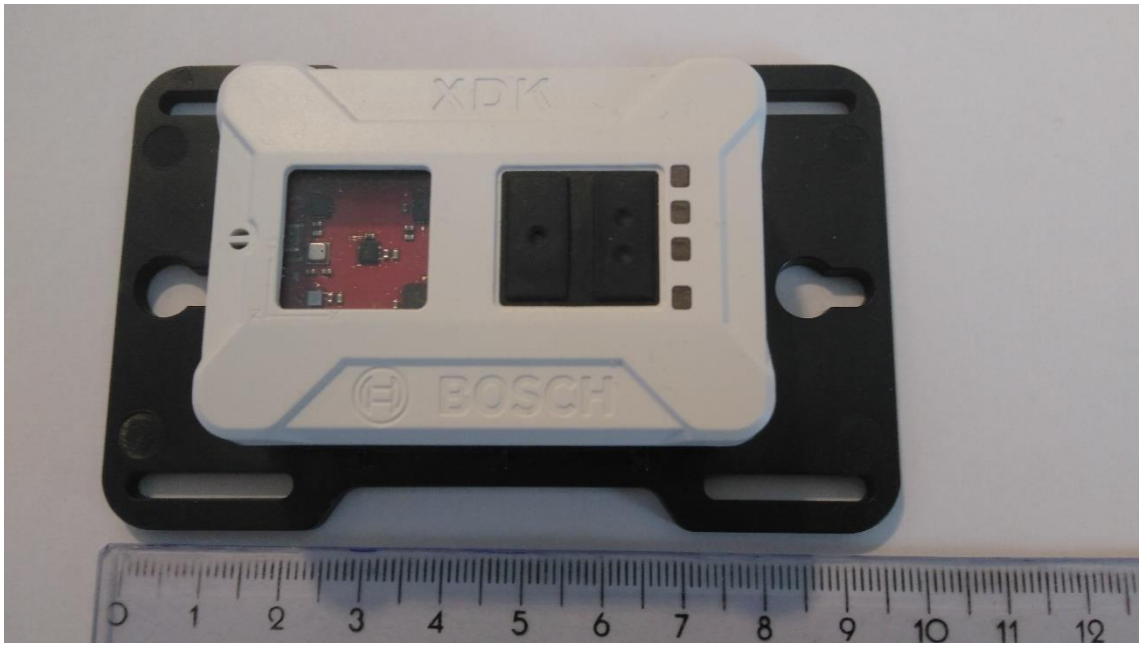
V další kapitole práce je uveden návrh aplikace pomocí blokového diagramu. Následně jsou navrženy varianty vyčítání dat, které odpovídají návrhu. Tyto varianty jsou dále zapracovány do navržených scénářů. V této kapitole je také ukázán postup pokusu řešení pomocí Windows knihoven a poznámky k řešení pomocí USB donglu s BLE toolkitem.

Celá práce je ukončena kapitolou s popisem vytvořeného programu a sepsány některé chyby, které se mohou objevit během chodu programu a navrženy možnosti jejich řešení.

1. BOSCH XDK 110

Bosch XDK 110 je programovatelné zařízení umožňující měření pomocí osmi vestavěných snímačů. Zařízení má vnitřní paměť. Je možné s ním komunikovat (a tedy i např. odesílat naměřená data pomocí USB, Bluetooth nebo Wi-Fi. Mimo vestavěných senzorů obsahuje také dvě programovatelná tlačítka označená jednou a dvěma tečkami a čtyři LED diody (zelená LED dioda pro signalizaci stavu nabíjení a programovatelné LED diody: červenou, oranžovou a žlutou). Pracovní podmínky jsou: teplota od –20 °C do 60 °C, vlhkost vzduchu od 10 % do 90 %, napájecí napětí 5 V DC.

XDK 110 je zařízení vhodné pro testování nových zařízení a technologií pro IoT a usnadňuje začátky produkce. Zařízení je vytvořeno tak, aby umožnilo rychle se připojit k jakémukoli elektronickému zařízení a spotřebiči, a to i k již vytvořenému zařízení. [11] a [12]



Obr. 1-1 Zařízení XDK 110

1.1 Vestavěné snímače

Bosch XDK 110 obsahuje osm snímačů, jmenovitě: akcelerometr, gyroskop, magnetometr, senzor intenzity osvětlení, teploměr, senzor tlaku, senzor vlhkosti a hlukový senzor.

Tab. 1-1 Základní údaje snímačů [1], [2] a [16]

Snímač	Rozsah	Vzorkovací frekvence
Akcelerometr	Programovatelný ±2 g až ±16 g	2 000 Hz (BMA280) 12,5 Hz - 1600 Hz (BMI160)
Gyroskop	Programovatelný ±125 °/s až ±2000 °/s	2 000 Hz (BMG160) 3200 HZ (BMI160)
Magnetometr	±1300 µT (osa X a Y) ±2500 µT (osa Z)	300 Hz
Senzor intenzity osvětlení	0,045 luxu až 188 000 luxu (22 bitů)	Není uveden
Teploměr	-20 °C až 60 °C	182 Hz
Tlakoměr	300 hPa až 1100 hPa	182 Hz
Měřič vlhkosti	10 % až 90 %	182 Hz
Zvukový snímač	60 Hz – 12,5 kHz	---

-

Rozdíl mezi akcelerometrem a gyroskopem BMI160 a akcelerometrem BMA280 a gyroskopem BMG160 je v tom, že BMI je inerciální snímač.

1.2 Komunikační rozhraní

Bosch XDK 110 umožňuje komunikovat přes rozhraní USB, pomocí kterého se dá i programovat. Dále umožňuje komunikovat na dálku pomocí Wi-Fi a již zmíněného BLE. Dále obsahuje konektor pro připojení rozšiřující desky "XDK Gateway", která rozšiřuje možnosti základní desky. XDK Gateway je ukázán na Obr. 1-2.

1.2.1 Rozdíly mezi komunikačními rozhraními Wi-Fi a BLE:

Rychlost – BLE je vhodné pro posílání menších bloků dat, které dokáže posílat rychlostí kolem 1 Mb/s, narozdíl od Wi-Fi, která je určena k posílání velkoobjemových dat a umožňuje je posílat rychlostí 347 Mb/s až 1,3 Gb/s

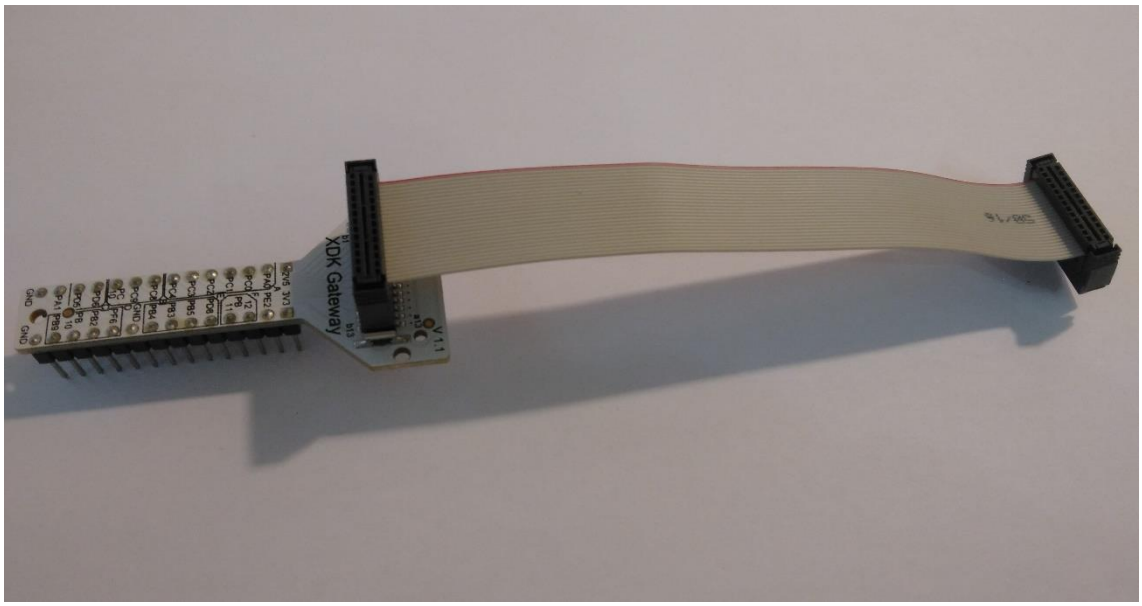
Bezpečnost – obě komunikační prostředí jsou v běžných případech dostatečně bezpečná, avšak pokud by se mělo jedna o choulostivá data je lepší využít rozhraní Wi-Fi,

které má již delší dobu zaimplementované bezpečnostní protokoly jako WEP, WPA a další novější protokoly.

Vzdálenost – dosah Bluetooth je obvykle mezi 40 m až 400 m, dosah Wi-Fi je až 92 metrů.

Kompatibilita – většina novějších zařízení dokáže komunikovat s nejnovějšími standardy jak Wi-Fi, tak Bluetooth, proto zde by problém být neměl. Toto však nemusí platit pro aplikace, se kterými chceme pracovat (například LabVIEW nepodporuje BLE, více k problematice v kapitole 2.4)

Spotřeba – ve většině případů mají zařízení BLE menší spotřebu jak zařízení Wi-Fi, a to i s relativně významným rozdílem. [16]

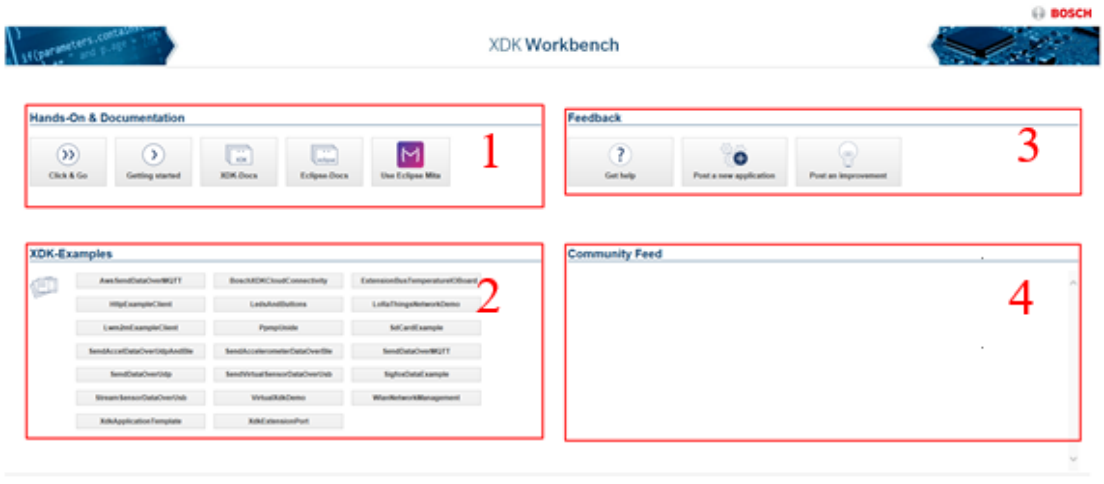


Obr. 1-2 Rozšiřující rozhraní XDK Gateway

1.3 XDK-Workbench

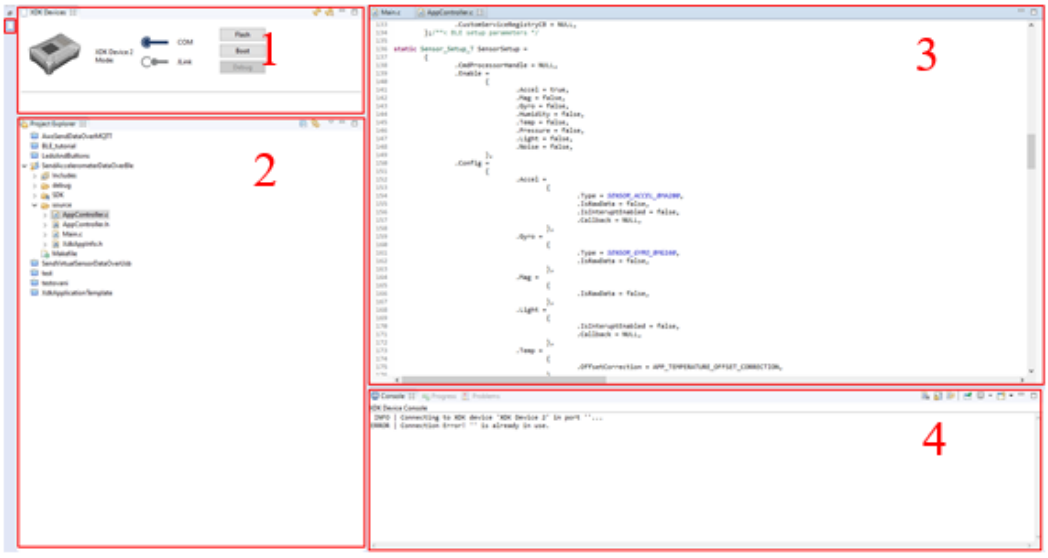
Aplikace XDK-Workbench slouží k programování zařízení Bosch XDK 110. XDK-Workbench umožňuje programovat v jazycích C, C++ a Mita. (Mita je jazyk zaměřující se na vytváření IoT a je určen zvláště pro vývojáře bez hlubších znalostí vývojového prostředí [10])

Úvodní obrazovka programu XDK-Workbench je na Obr. 1-3. Rámeček č. 1 označuje skupinu příkazů pro práci s projektem a s dokumenty. V rámečku č. 2 jsou vzorové projekty pro XDK, v rámečku č. 3 jsou ikony pro komunikaci se společností Bosch a v rámečku č. 4 lze nalézt odkazy na čerstvá témata a otázky na fóru XDK.



Obr. 1-3 Úvodní obrazovka XDK-Workbench

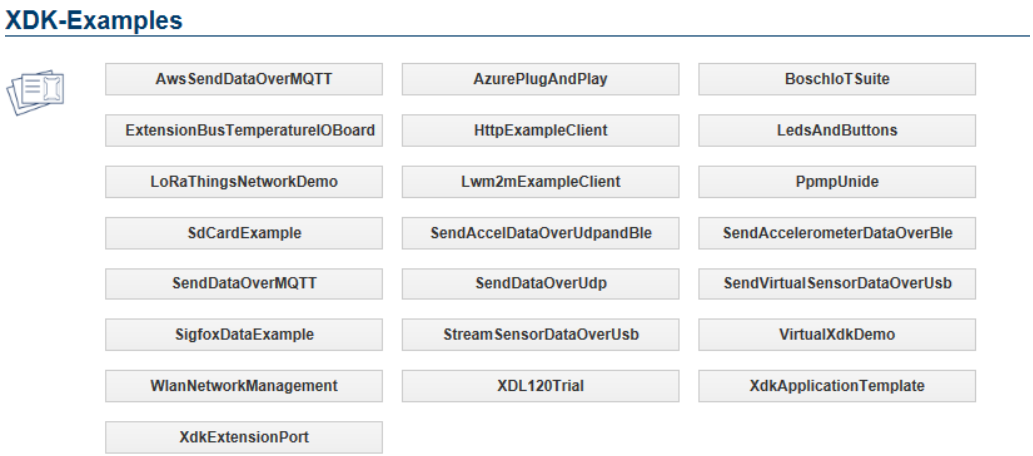
Na Obr. 1-4 je obrazovka po vytvoření projektu. Ikonka XDK na levé liště (označená na obrázku malým červeným rámečkem) umožní návrat na úvodní obrazovku programu. V rámečku č. 1 se nachází seznam připojených zařízení, zde je vidět, zda je zařízení aktuálně připojeno. Zařízení je zde možno přejmenovat, případně do něj nahrát napsaný kód. V rámečku č. 2 jsou vidět projekty, které jsou nebo byly používány. Dále je zde vidět, zda je projet uzavřen nebo otevřen a u otevřených projektů je zobrazena jejich struktura. V rámečku č. 3 se objeví po otevření zdrojového kódu programovací list, do kterého lze vepsat program pro XDK. Rámeček č. 4 zobrazuje konzoli, ve které se nacházejí informace o překladu a nahrávání programu, dále pak také změřené hodnoty při komunikaci přes USB. V sekci Window → Perspective → Customize Perspective lze obrazovku po vytvoření projektu různě upravovat podle vlastních preferencí. Proto vzhled, který je na Obr. 1-4 nemusí být pro každého uživatele stejný.



Obr. 1-4 Obrazovka při prvním otevření projektu

1.4 Příklady v XDK-Workbench

XDK-Workbench obsahuje mnoho příkladů (examples) Všechny tyto ‚examply' jsou vidět na Obr. 1-5 Examply v XDKObr. 1-5. Během této bakalářské práce byl použit example ‚SendAccelDataOverUdpandBLE'. Mimo tento example byl použit také example ‚SendVirtualSensorDataOverUSB' pro otestování nahrávání do XDK a práce s XDK-Workbench. Všechny examply jsou složeny z ‚main' souboru, souboru aplikace a mnoha dalších hlavičkových souborů. Pro samotnou práci s examply je nejdůležitější hlavně soubor AppControler, který obsahuje popis a kód jednotlivých funkcí. Avšak setkal jsem se i s informacemi, které se navzájem vylučovaly. Například v popisku celého examplu je uvedeno, že příkaz na zapnutí odesílání dat je 7374617274_{HEX}, ale v celém kódu je nastaven příkaz na 1_{DEC}. V práci jsem zkoušel obě možnosti a ani při jedné se mi nepovedlo vyčíst data z XDK, takže je možné, že někde v kódu jsou tyto hodnoty ještě přepsány jinými hodnotami, což se mi dosud nepodařilo objevit.



Obr. 1-5 Examply v XDK

Pro práci s examplem je důležitější si přečíst úvodní komentář v souboru AppControler obsahující důležité informace o fungování examplu, případně i parametry, které je potřeba nastavit pro začátek posílání dat a například i informace o tom, kde se nachází měřená data a kam naopak zapisovat data.

```
/**
 * @ingroup APPS_LIST
 *
 * @defgroup SEND_ACCELEROMETER_DATA_OVER_BLE SendAccelerometerDataOverBle
 * @{
 *
 * @brief Demo application of Transmitting BMA280 Accelerometer data on BLE(Bluetooth Low Energy) every configured interval (#APP_CONTROLLER_TX_DELAY)
 *
 * @details This example demonstrates how to read sensor values from the BMA280 Acceleration sensor and streams them over Bluetooth Low Energy via custom Bi-Directional Service.<br>
 * Either use your Android or iOS mobile phone (see [Android](https://play.google.com/store/apps/details?id=com.macdom.ble.blescanner&hl=en) or
 * [iOS](https://itunes.apple.com/in/app/lightblue-explorer-bluetooth-low-energy/id557428110?mt=8) App Store) to connect to XDK and receive the data.
 * Send command <b>start (HEX: 0x7374617274)</b>  to XDK via Bluetooth Low Energy, so that streaming of data begins.
 * To stop the streaming send command <b>end (HEX: 656e64)</b>
 *
 * This Application enables the bi-directional service in ble and sends Accelerometer Data over ble . <br>
 * <b> Bi-Directional Service : </b>
 *
 *  Service             |  Characteristic               |        Attribute-Type             |             UUID                    |
 * ---------------------|-------------------------------|-----------------------------------|-------------------------------------|
 * BidirectionalService |    NA                         |        uint8_t                    | b9e875c0-1cfa-11e6-b797-0002a5d5c51b |
 * NA                   |    Rx                         |        uint8_t X[20]              | 0c68d100-266f-11e6-b388-0002a5d5c51b |
 * NA                   |    Tx                         |        uint8_t X[20]              | 1ed9e2c0-266f-11e6-850b-0002a5d5c51b |
 *
```

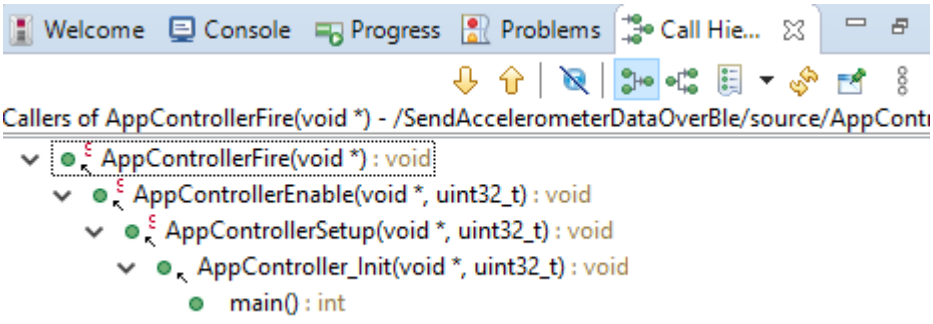
Obr. 1-6 Ukázka popisu examplu

Mimo tento popis obsahuje soubor AppControler definice proměnných a následně jednotlivé funkce obsahující samotný kód programu. Ukázka funkce používané pro odesílání dat přes Bluetooth je uvedena na Obr. 1-7.

```
static void AppControllerFire(void* pvParameters)
{
    BCDS_UNUSED(pvParameters);

    Retcode_T retcode = RETCODE_OK;
    Sensor_Value_T sensorValue;
    uint8_t accelDataBleTxBuffer[APP_CONTROLLER_BLE_TX_LEN] = { 0 };

    memset(&sensorValue, 0x00, sizeof(sensorValue));
    while (1)
    {
        if (AppControllerBleTransmitPayload && BLE_IsConnected())
        {
            /* Resetting / clearing the necessary buffers / variables for re-use */
            retcode = RETCODE_OK;
            memset(accelDataBleTxBuffer, 0U, sizeof(accelDataBleTxBuffer));

            retcode = Sensor_GetData(&sensorValue);
            if (RETCODE_OK == retcode)
            {
                sprintf((char*) accelDataBleTxBuffer, "%ld %ld %ld", (long int) sensorValue.Accel.X, (long int) sensorValue.Accel.Y, (long int) sensorValue.Accel.Z);
                /* printing Accel data of BMA280 on serialport for validation */
                printf("BMA280 Accel Data in  milli G :\n\rx =%ld\n\ry =%ld\n\rz =%ld\r\n",
                        (long int) sensorValue.Accel.X,
                        (long int) sensorValue.Accel.Y,
                        (long int) sensorValue.Accel.Z);
            }
            if (RETCODE_OK == retcode)
            {
                memset(accelDataBleTxBuffer, 0x11, sizeof(accelDataBleTxBuffer));
                retcode = BLE_SendData((uint8_t*) accelDataBleTxBuffer, (uint8_t) sizeof(accelDataBleTxBuffer), NULL, APP_CONTROLLER_BLE_SEND_TIMEOUT_IN_MS);
            }
            if (RETCODE_OK != retcode)
            {
                Retcode_RaiseError(retcode);
            }
        }
        vTaskDelay(pdMS_TO_TICKS(APP_CONTROLLER_BLE_TX_DELAY));
    }
}
```

Obr. 1-7 Ukázka funkce v examplu

Tyto funkce jsou navzájem hodně provázané proto je dobré pro zjištění přesných volání použít příkaz Call Hierarchy. Ta se dá otevřít po označení funkce kliknutím na pravé tlačítko myši a poté levým tlačítkem na Open Call Hierarchy. Ukázka vzhledu je na Obr. 1-8.



Obr. 1-8 Ukázka Call Hierarchy

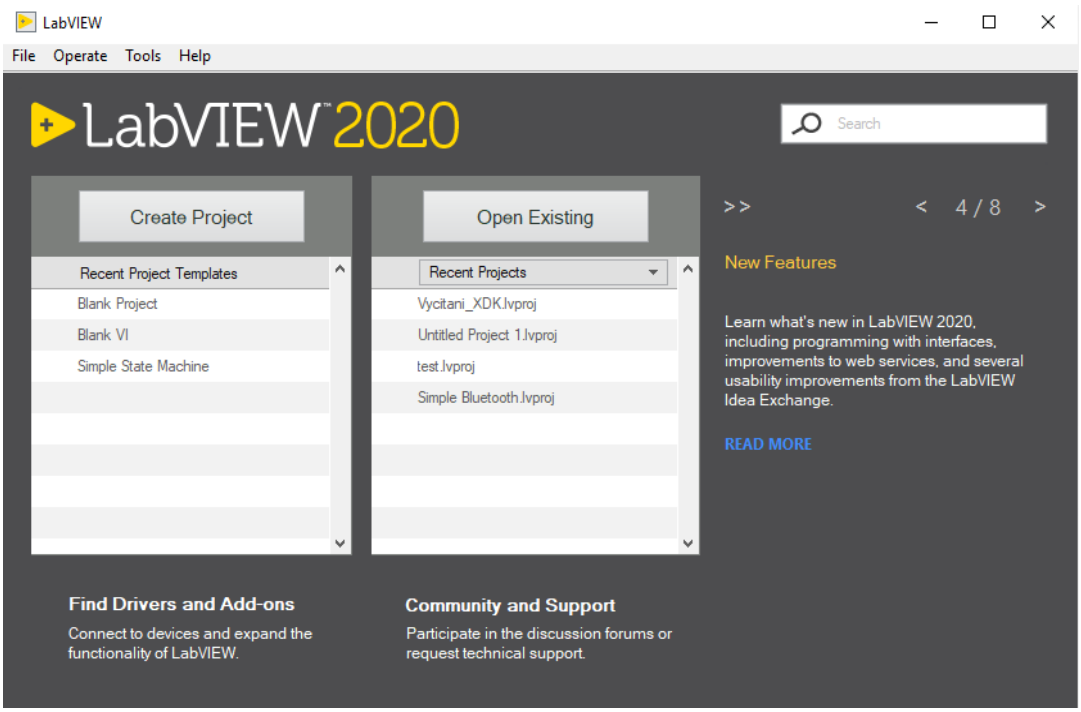
2. KOMUNIKACE LABVIEW SE ZAŘÍZENÍM STANDARDU BLE

2.1 LabVIEW

LabVIEW je programovací a vývojářské prostředí od americké firmy National Instruments. Při vytváření aplikací se v LabVIEW využívá grafického programování, které je vhodné nejen pro měření a analýzu signálů, ale lze ho také použít při programování složitých systémů jako je třeba řízení robota. Hlavním cílem virtuální instrumentace je nahradit hardware za softwarové řešení, které je ve značné míře flexibilnější a levnější [13].

Vývoj LabVIEW začal v roce 1983 po uvedení desky pro rozhraní GPIB od firmy National Instrument. Za ‘otce LabVIEW’ je považován Jeffrey Kodosky, který zahájil vývoj grafického vývojového nástroje, na kterém LabVIEW staví. Hlavním předpokladem bylo, aby technik, který umí zapsat své poznatky a požadavky do blokového diagramu, dokázal samostatně a stejným způsobem napsat i program [13].

Při spuštění aplikace se zobrazí úvodní obrazovka LabVIEW. Po kliknutí na ‚Create Project‘ můžeme vytvořit nový projekt, a to nejen prázdný, ale je zde i mnoho šablon a vzorových projektů, ze kterých můžeme vycházet. Po kliknutí na ‚Open Existing‘ můžeme otevřít již vytvořený projekt.



Obr. 2-1 Úvodní obrazovka LabVIEW

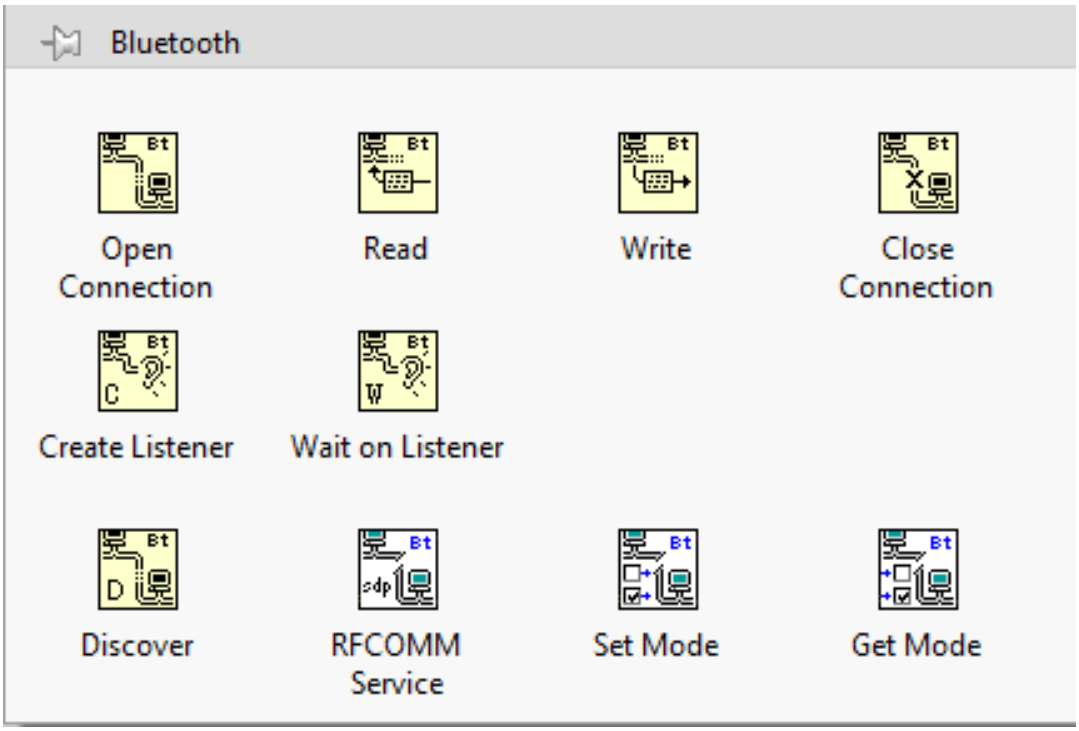
Každá aplikace VI (Virtual Instrument, prvek programu LabVIEW) se skládá ze dvou oken, a to Čelního panelu a Blokového diagramu. Čelní panel slouží jako uživatelské rozhraní k vypracované aplikaci. Blokový diagram je místo pro programátora aplikace, kde lze pomocí funkčních bloků poskládat kód programu. Mimo bloků, které již obsahuje samotné LabVIEW, lze využít také uživatelské knihovny, případně si vytvořit vlastní subVI.

2.2 Komunikace LabVIEW přes Bluetooth

LabVIEW má v sobě vloženu knihovnu pro komunikaci přes Bluetooth. Tuto knihovnu lze najít po otevření výběrového menu na blokovém diagramu, kliknutím pravým tlačítkem myši v blokovém diagramu, následně v záložce Data Communication → Protocols → Bluetooth.

Základem komunikace s LabVIEW přes Bluetooth jsou čtyři VI ukázané na Obr. 2-2 v prvním řádku, a to prvky Open Connection, Read, Write a Close Connection. Prvky Open Connection / Close Connection slouží ke spuštění / ukončení komunikace se zařízením. Prvek Read slouží k vyčítání změřených hodnot. Prvek Write slouží k nastavení potřebných vlastností zařízení.

Tato volba však neumožňuje komunikovat se zařízeními, které používají technologii BLE, jelikož v aktuální verzi (2020) není standard BLE podporován [6]. Jak lze tento problém vyřešit je uvedeno v kapitole 2.4 Komunikace LabVIEW přes BLE.



Obr. 2-2 Knihovna funkcí pro komunikaci s Bluetooth

2.3 Co je to BLE

Následující informace jsou čerpány ze zdrojů [8] a [9]. Bluetooth Low Energy je bezdrátová komunikační technologie s nízkou spotřebou energie. Je to poměrně nový standard, který vznikl kolem roku 2009 pro použití v zařízeních s nízkou spotřebou. Pro Bluetooth Low Energy se používá mnoho označení jako například Bluetooth LE, Bluetooth Smart, či zkratka BLE. Standard byl původně navržen společností Nokia pod názvem Wibree.

Zařízení BLE jedná buď v "centrální" nebo v "periferní" roli, kdy centrální rolí je myšlen klient, zařízení, které iniciuje příkazy a požadavky, např. počítač, a periferní rolí je myšlen server, zařízení, které přijímá příkazy a požadavky, např. XDK. Tato komunikace je označována jako online režim a používá se pro ni protokol GATT. Tento protokol definuje, jak dvě zařízení vybavená BLE mezi sebou komunikují.

GATT (**G**eneric **Att**ribute Profile) definuje způsob, jakým dvě zařízení BLE mezi sebou komunikují pomocí konceptů zvaných 'services' a 'charakteristics'. Používá se protokol s názvem Attribute protocol (ATT), který využívá k ukládání dat jednoduchou tabulku obsahující služby, charakteristiky a související data [14].

Následující odstavce do konce podkapitoly jsou ze zdroje [17]. Nejmenší datová jednotka definovaná pro ATT je atribut. To je adresovatelná informace, která může obsahovat různá uživatelská data. ATT dokáže pracovat pouze s atributy, a proto všechny informace musí být organizovány v této podobě. Atributy jsou vždy uloženy na serveru, takže práce s ATT nevyžaduje žádnou interní paměť. Každý atribut obsahuje informace o sobě samém a poté skutečná data v polích. K popsání jednotlivých atributů se používá takzvaný Attribute Handle, zkráceně Handle. To je hexadecimální číslo mezi 0001 až FFFF.

Mimo Handle se při komunikaci přes GATT používají ještě dvě další specifikace, a to typ a oprávnění. Typ je UUID, což je jedinečný identifikátor tak velký, aby nebyl opakovatelný, používají se 16, 32 nebo 128bitové UUID a určují druh dat, který je uložen v hodnotě atributu. Ačkoli typ je vždy UUID, tak lze použít mnoho druhů UUID, jako například servisní, charakteristické a profilové. Druhou specifikací je oprávnění, to jsou metadata, která určují, co lze s atributem dělat a s jakými bezpečnostními požadavky.

Přístupová oprávnění:

- Žádné (None) – nelze číst ani zapisovat
- Readable – lze číst
- Writable – lze zapisovat
- Readable and writable – Lze číst a zapisovat

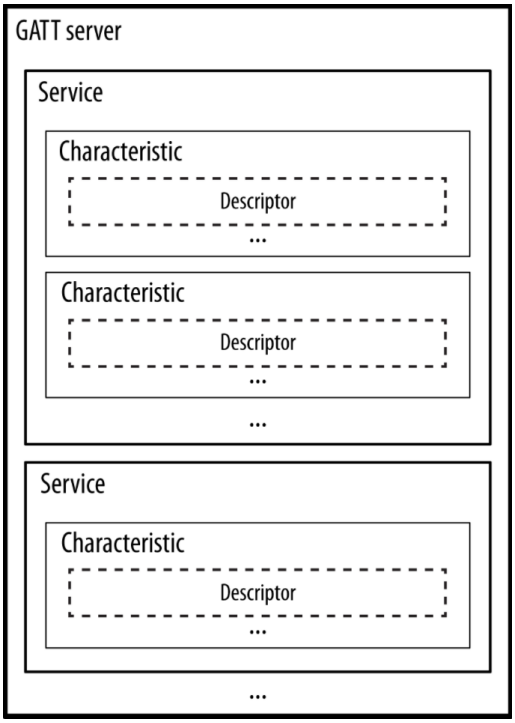
Šifrování:

- Není vyžadováno
- Neověřené šifrování – musí být šifrováno, ale šifrovací klíče není třeba ověřovat

- Ověřené šifrování – pro připojení musí být šifrováno autorizovaným klíčem

Autorizace:

- Vyžadována autorizace
- Není vyžadována autorizace



Obr. 2-3 Datová hierarchie GATT [17]

Pokud tyto atributy vložíme do tabulky, vznikne nám tabulka podobná tabulce na Obr. 2-4. Pokud bychom tuto tabulku lehce upravili, ‚Type' nahradíme za dvě UUID – servisní a charakteristické a ‚Permissions', ‚Value' a ‚Value lenght' je definované přímo v samotných datech na začátku celého pole.

Handle	Type	Permissions	Value	Value length
0x0201	$UUID_1$ (16-bit)	Read only, no security	0x180A	2
0x0202	$UUID_2$ (16-bit)	Read only, no security	0x2A29	2
0x0215	$UUID_3$ (16-bit)	Read/write, authorization required	"a readable UTF-8 string"	23
0x030C	$UUID_4$ (128-bit)	Write only, no security	{0xFF, 0xFF, 0x00, 0x00}	4
0x030D	$UUID_5$ (128-bit)	Read/write, authenticated encryption required	36.43	8
0x031A	$UUID_1$ (16-bit)	Read only, no security	0x1801	2

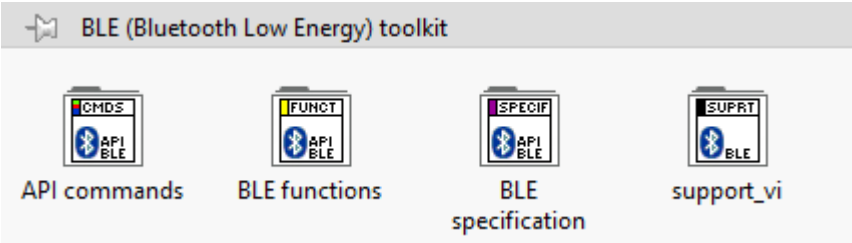
Obr. 2-4 Atributy vložené do tabulky [17]

2.4 Komunikace LabVIEW přes BLE

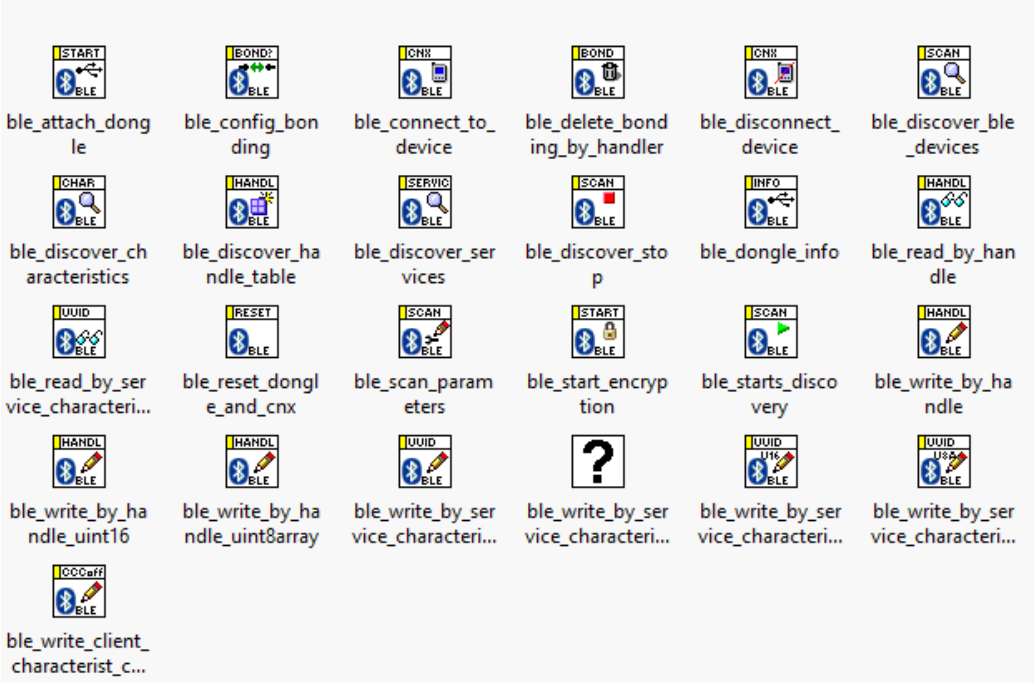
LabVIEW v sobě neobsahuje žádné knihovny, které by umožnily komunikaci přes standard BLE [6], na rozdíl od klasického Bluetooth, a proto nelze bez úpravy využít komunikaci přes BLE. Problém lze řešit dvěma způsoby, a to pomocí nástroje vytvořeného komunitou programátorů LabVIEW nebo použitím knihovních funkcí Windows. Na žádnou z těchto variant však nelze použít základní vestavěné primitivní funkce LabVIEW, založené na starších ovladačích Windows, které BLE nepodporují.

2.4.1 Řešení pomocí nástroje od komunity

Možnost komunikace pomocí komunitou vytvořené knihovny pro práci s BLE zařízeními v LabVIEW [7], přesněji toolkit vytvořila společnost Silicon Lab. Při použití tohoto programového nástroje je potřeba převodník z USB na Bluetooth (USB dongle), který změní tuto komunikaci tak, že nebude pro LabVIEW nutné komunikovat přes Bluetooth. Bude totiž komunikovat pouze s USB, kdy toolkit bude upravovat požadavky do podoby takové, aby USB dongle dokázal zprávy převést na komunikaci přes Bluetooth. Využitelné funkce jsou na Obr. 2-5 a podsložce Obr. 2-6.



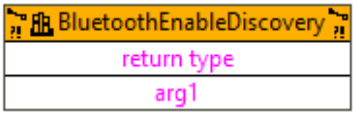
Obr. 2-5 Složka pro BLE toolkit



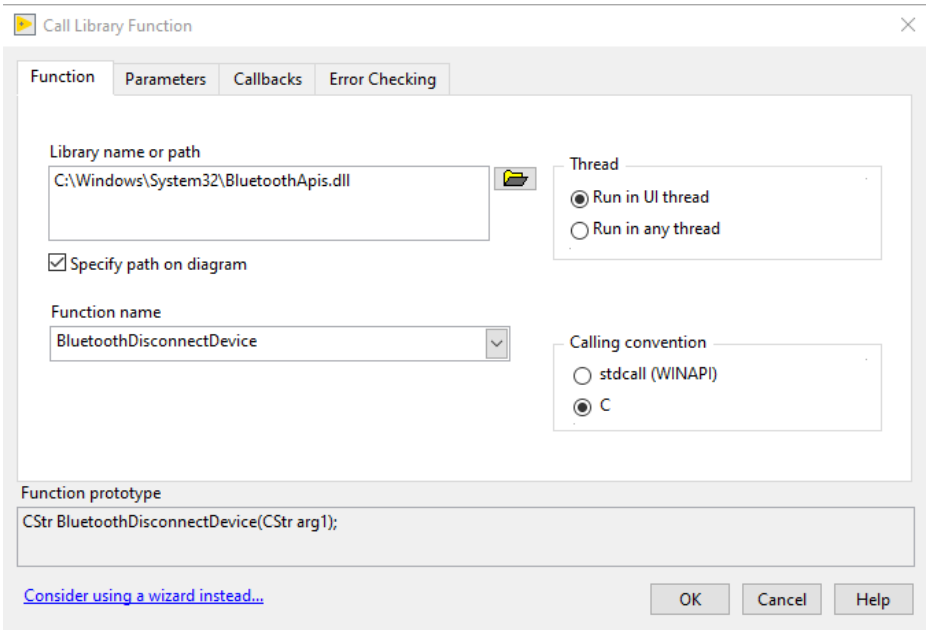
Obr. 2-6 Funkce BLE toolkitu

2.4.2 Řešení pomocí knihoven OS Windows

Další možností je využít knihovnu v rámci OS Windows [6]. Toto řešení je možné pouze pro systémy Windows 8 a vyšší, které obsahují knihovnu se standardem Bluetooth 4.0 a vyšším. K tomuto musíme využít funkci “Call Library Function Node”, která slouží pro vložení kódu z knihovny do LabVIEW. (Obr. 2-7)

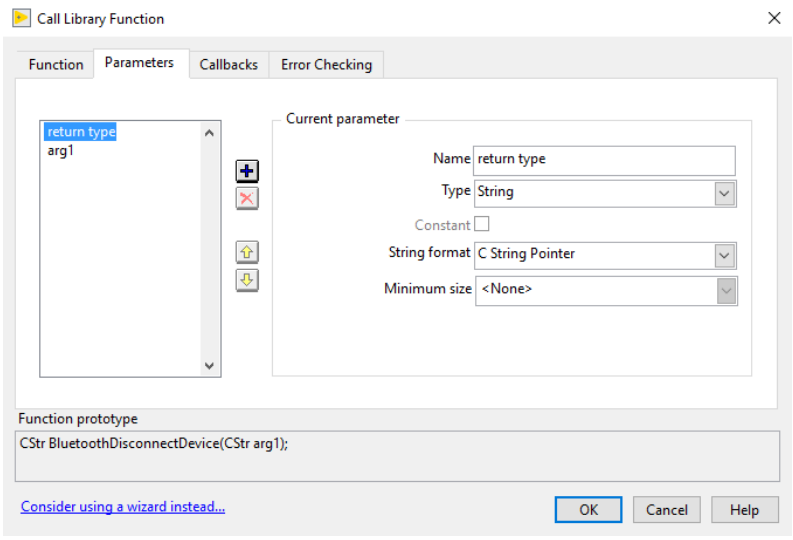


Obr. 2-7 Call Library Function Node



Obr. 2-8 Obrazovka nastavení Call Library Function Node

Na Obr. 2-7 je ukázána funkce „Call Library Function Node“, tak jak vypadá na Blokovém diagramu. Na Obr. 2-8 je ukázána po rozkliknutí jejího nastavení. Do kolonky Library name or path se vloží cesta ke knihovně, nebo pouze název knihovny, pokud je cesta definována pomocí vstupu ‚path‘, který se zobrazí při zakliknutí checkboxu pod oknem s názvem souboru. Do této kolonky si nastavíme adresu k souboru knihovny BluetoothApis.dll, která obsahuje funkce pro komunikaci s BLE. Kolonka function name určuje, kterou funkci z knihovny chceme využít a kolonka Function prototype ukazuje funkci i se vstupy a výstupy, které jsou aktuálně nastaveny do funkce Call Library Function. Na Obr. 2-9 je vidět obrazovka vstupů a výstupů funkce Call Library Function, tyto vstupy je potřeba nastavit podle požadované funkce.

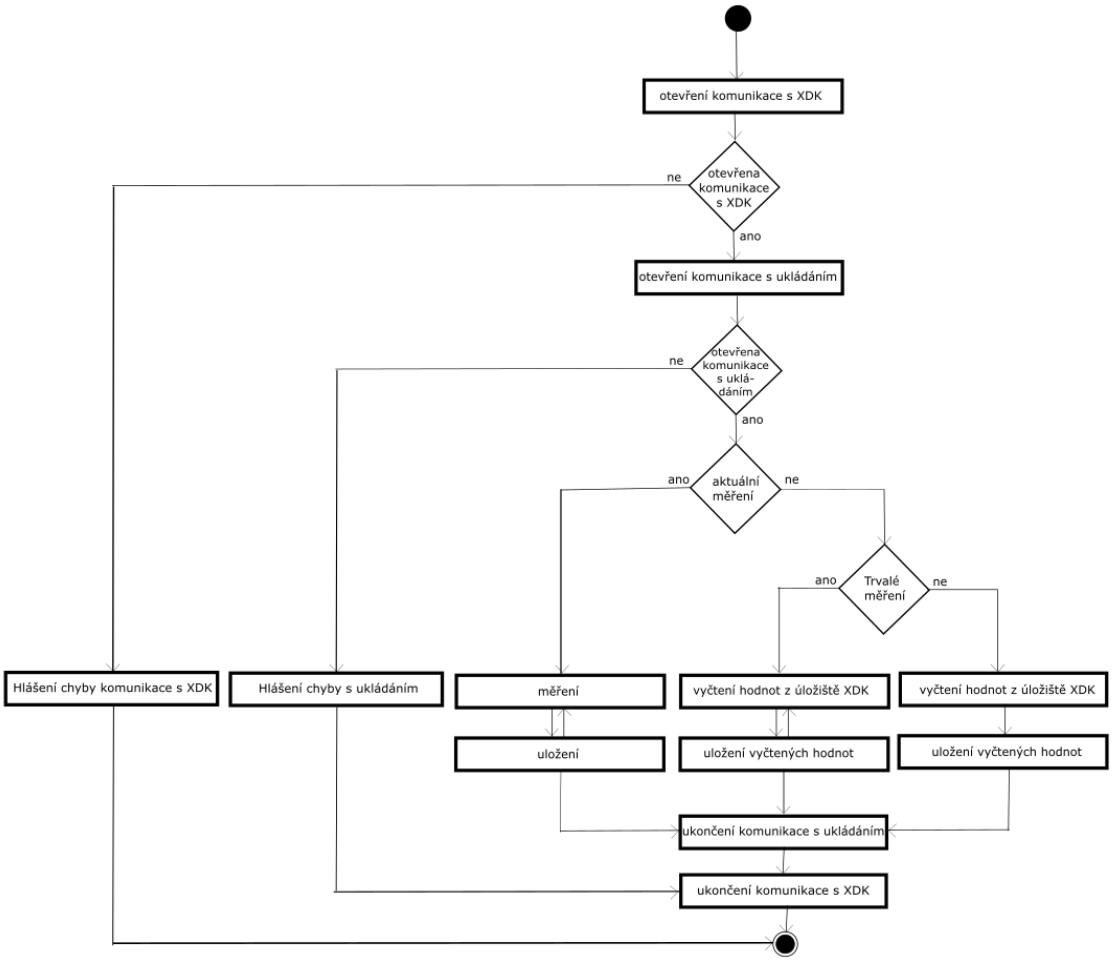


Obr. 2-9 Obrazovka nastavení vstupů a výstupů

3. NÁVRH APLIKACE V LABVIEW

3.1 Čtení z XDK

Návrh aplikace počítá s tím, že aplikace bude obsahovat tři varianty vyčítání dat z XDK „Aktivní čtení“, „Blokové čtení“ a „Jednorázové čtení“. Schéma je uvedeno na Obr. 3-1.



Obr. 3-1 Schéma návrhu programu pro LabVIEW

3.1.1 Aktivní čtení

Aktivní čtení vyčítá z XDK vždy jednu poslední změřenou hodnotu ze sledované veličiny. Na Obr. 3-1 se aktivní čtení nachází v levé větvi schématu. Při spuštění se nejdříve otevře komunikace s XDK, potom se nastaví komunikace se souborem, do kterého se budou data ukládat. V případě, že soubor ještě nebyl vytvořen, vytvoří se, v případě, že již soubor bude existovat, soubor se buď přepíše, nebo se záznam přidá za stávající data. Poté se zapne nekonečná měřící smyčka, která vždy vyčte údaje z XDK a následně je uloží. V případě, že si uživatel vyžádá vypnutí měření, program přejde na

uzavření souboru, do kterého jsou vkládány změřené hodnoty, a následně ukončí komunikaci s XDK.

Popis jednoho vyčtení: Poslední hodnota od každé měřené veličiny.

3.1.2 Blokové čtení

Blokové čtení vyčítá z XDK vždy blok hodnot sledovaných veličin uložený v paměti XDK. Na Obr. 3-1 se blokové čtení nachází v prostřední větvi schématu. Při spuštění se nejdříve otevře komunikace s XDK, potom se nastaví komunikace se souborem, do kterého se budou data ukládat. V případě, že soubor ještě nebyl vytvořen, vytvoří se, v případě, že již soubor bude existovat, soubor se buď přepíše, nebo se záznam přidá za stávající data. Poté se zapne nekonečná vyčítací smyčka, která v každém cyklu vyčte blok dat z paměti XDK a následně je uloží. V případě, že si uživatel vyžádá vypnutí vyčítání, program přejde na uzavření souboru, do kterého jsou vkládány změřené hodnoty, a následně ukončí komunikaci s XDK.

Popis jednoho vyčtení: Blok hodnot uložených v paměti s daty všech měřených veličin.

3.1.3 Jednorázové čtení

Jednorázové čtení vyčítá z XDK právě jeden blok hodnot sledovaných veličin uložených v paměti XDK. Na Obr. 3-1 se jednorázové čtení nachází v pravé větvi schématu. Při spuštění se nejdříve otevře komunikace s XDK, potom se nastaví komunikace se souborem, do kterého se budou data ukládat. V případě, že soubor ještě nebyl vytvořen, vytvoří se, v případě, že již soubor bude existovat, soubor se buď přepíše, nebo se záznam přidá za stávající data. Poté se vyčte blok dat z paměti XDK a uloží se. Následně program uzavře soubor a ukončí komunikaci s XDK.

Popis jednoho vyčtení: Blok hodnot uložených v paměti s daty všech měřených veličin.

3.2 Scénáře pro posílání dat

Pro aplikaci navrhuji následujících devět scénářů pro posílání dat mezi XDK a LabVIEW, které vycházejí z výše zmíněných variant vyčítání dat z XDK:

1) Aktivní čtení, aktivované sensory teploty, tlaku a vlhkosti
2) Blokové čtení, aktivované sensory teploty, tlaku a vlhkosti
3) Jednorázové čtení, aktivované sensory teploty, tlaku a vlhkosti
4) Aktivní čtení, aktivovaný gyroskop a akcelerometr
5) Blokové čtení, aktivovaný gyroskop a akcelerometr
6) Jednorázové čtení, aktivovaný gyroskop a akcelerometr
7) Aktivní čtení, aktivovaný senzor intenzity osvětlení

8) Blokové čtení, aktivovaný senzor intenzity osvětlení
9) Jednorázové čtení, aktivovaný senzor intenzity osvětlení

3.3 Pokus řešení přes Windows knihovny

První pokus vytvořit aplikaci byl pomocí Windows knihoven. Stručný popis tohoto řešení je v kapitole 2.4.2 Řešení pomocí knihoven OS Windows. Toto řešení se však nevedlo k výsledku, protože na internetu jsem nenašel ideální popis všech funkcí. V literatuře [15] je mnoho popisů funkcí, ale chybí popisy některých funkcí jako je například popis funkce ‚Disconnect'. Pro zprovoznění aplikace je však nutná minimálně znalost vstupů a výstupů. Proto jsem se rozhodnul tímto způsobem bakalářskou práci neřešit a zvolil raději variantu pomocí komunitního toolkitu.

3.4 Poznatky k řešení pomocí komunitního toolkitu

Pro řešení problému jsem proto nakonec zvolil variantu toolkitu umožňující komunikaci s BLE zařízeními pomocí USB donglu.

3.4.1 Reset

Protože při vytváření programu se objevovala chyba, která zablokovala komunikaci USB a následně bylo nutné restartovat celý počítač, byl na začátek programu přidán příkaz ‚Reset USB'. Tento reset odstraní případné zablokování USB konektoru z důvodu jeho předchozího obsazení. Toto následně zajistí bezchybný chod programu.

3.4.2 Přepnutí XDK mezi bootingem a aplikací

V případě, že komunikujete s XDK (případně před spuštěním komunikace), např pomocí LabVIEW, není dobré spouštět aplikaci XDK-Workbench. Při jejím spuštění se XDK přepne do režimu bootování, čímž se vypne aktuálně běžící program. To způsobí, že se vypne komunikace a LabVIEW ohlásí chybu, že komunikační zařízení BLE není v XDK zapnuto. Řešení je uvedeno v kapitole 4.2.2.

3.4.3 Úprava UUID z XDK-Workbench do LabVIEW

Pro komunikaci přes Bluetooth je nutné znát UUID, to se dá pro example, který byl použit, zjistit v komentářích v programu, tyto UUID však nelze přímo použít v LabVIEW, je potřeba je prvně upravit. UUID z XDK-Workbench obsahuje pomlčky, ty je nutno odebrat a také obsahuje malá písmena, ta je nutno přepsat na velká. Například z examplu přečteme, že Service UUID je b9e875c0-1cfa-11e6-b797-0002a5 ale do LabVIEW musíme zapsat B9E875C01CFA11E6B7970002A5. Tyto UUID lze najít také v Handle tabulce.

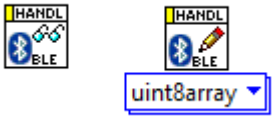
3.4.4 Handle tabulka

Handle table

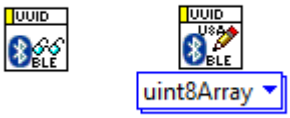
0	0001	1801	2800
0	0002	1801	2803
	0003	1801	2A05
	0004	1801	2902
	0005	1800	2800
	0006	1800	2803
	0007	1800	2A00
	0008	1800	2803
	0009	1800	2A01
	000A	1800	2803
	000B	1800	2A28
	000C	1800	2803
	000D	1800	2A29
	000E	B9E875C01CFA11E6B7970	2800
	000F	B9E875C01CFA11E6B7970	2803
	0010	B9E875C01CFA11E6B7970	0C68D100266F11E6B38800
	0011	B9E875C01CFA11E6B7970	2803
	0012	B9E875C01CFA11E6B7970	1ED9E2C0266F11E6850B00

Obr. 3-2 Handle tabulka

Handle tabulka je základem pro komunikaci pomocí Bluetooth. Vzhled Handle tabulky je ukázán na Obr. 3-2. Každý řádek obsahuje informace poskytované XDK, do jednotlivých řádků můžeme zapisovat a vyčítat pomocí toolkitu dvěma způsoby, pomocí handlu, nebo za pomoci dvou UUID.



Obr. 3-3 VI z toolkitu se čtením a zápisem pomocí handlu



Obr. 3-4 VI z toolkitu se čtením a zápisem pomocí UUID

Pro vyčítání pomocí handlu jsou na Obr. 3-3 ukázány funkce, které umožnují vyčítat, nebo zapisovat informace pomocí handlu. Tento handle najdeme v Handle tabulce v prvním sloupečku. Druhým způsobem čtení je pomocí dvou UUID, kterými jsou Charakteristické UUID a Servisní UUID. Servisní UUID se dá v Handle tabulce najít ve druhém sloupečku. Charakteristické UUID se v Handle tabulce nachází v třetím sloupečku.

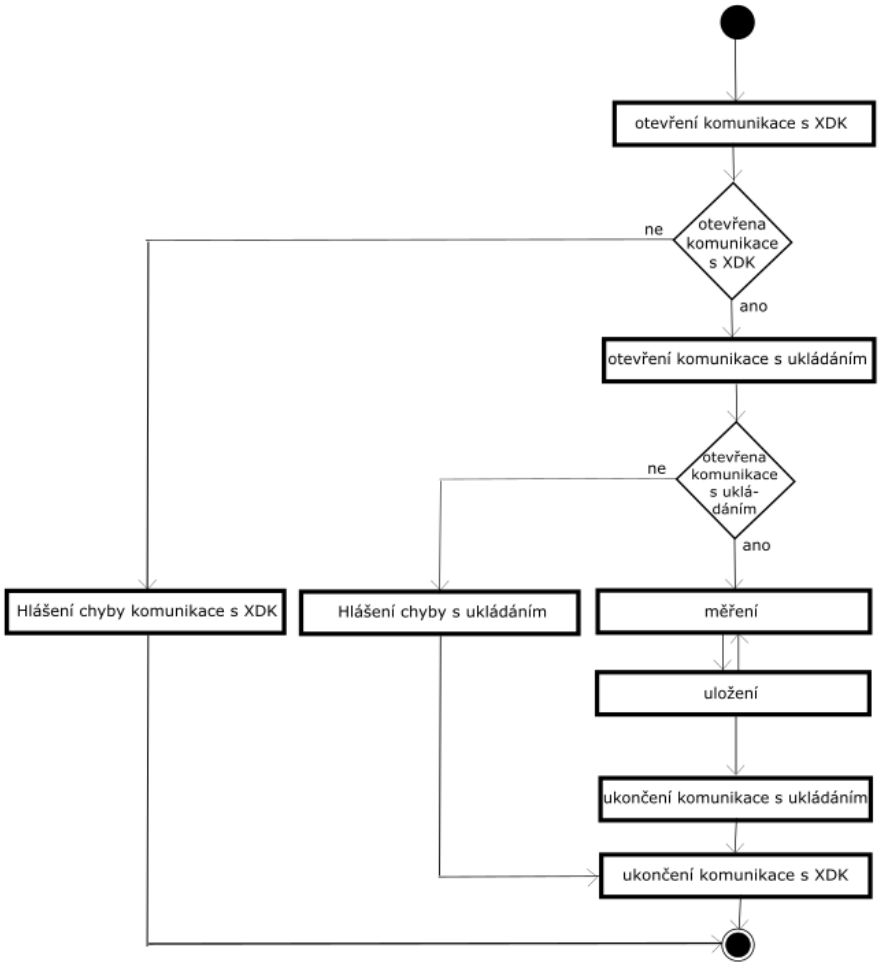
3.4.5 Globální proměnné

V kódu byly využity globální proměnné pro zpřehlednění kódu, avšak mělo by být pomocí ‚case' zajištěno přesné pořadí toho, kdy se co vyčte. Takže by neměla hrozit nějaká chyba nebo pád programu spojený s vyčtením či zapsáním do proměnné ve špatném okamžiku.

4. POPIS APLIKACE V LABVIEW

4.1 Aplikace

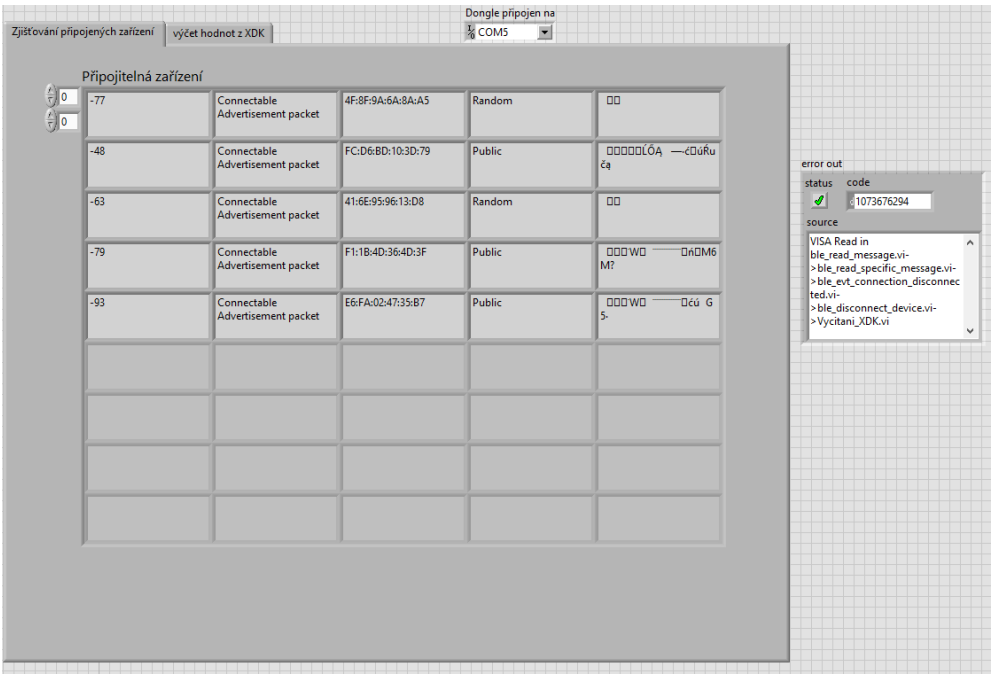
Původním plánem bylo vytvořit devět scénářů pro vyčítání, to se však kvůli problémům s řešením komunikace přes BLE nepovedlo zrealizovat. Proto jsem se rozhodl pro realizaci jednoho scénáře, ve kterém se posílají aktivně hodnoty z akcelerometru a ty se následně vyčítají. Struktura programu je uvedena na Obr. 4-1. Aplikace má stále problémy s vyčítáním, ač se zdá, že komunikace mezi XDK a aplikací funguje v pořádku. Aplikace v XDK hlásí, že měřené hodnoty jsou nulové. Tento problém může být způsoben protichůdnými informacemi v kódu, kde v popisku je psáno, že pro začátek má být poslán do XDK příkaz 7374617274_{HEX}, ale při hledání příkazu v kódu jsem našel, že příkaz je nastaven 1_{DEC}. V práci jsem zkoušel obě možnosti a ani při jedné se mi nepovedlo vyčíst data z XDK, takže je možné, že někde v kódu jsou tyto hodnoty ještě přepsány jinými hodnotami, což se mi dosud nepodařilo objevit.



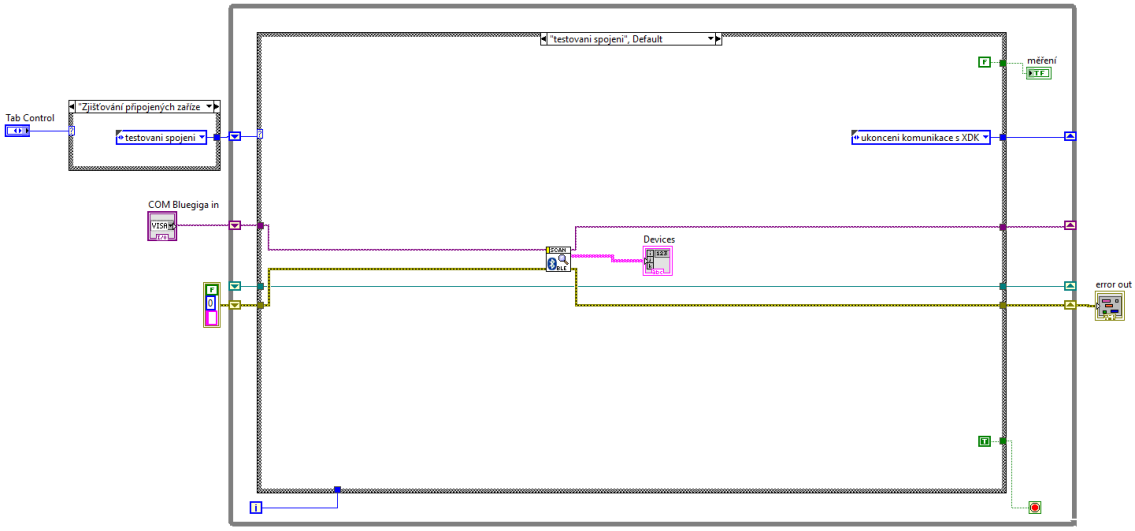
Obr. 4-1 Aktuální struktura programu

4.1.1 Volba zjišťování připojených zařízení

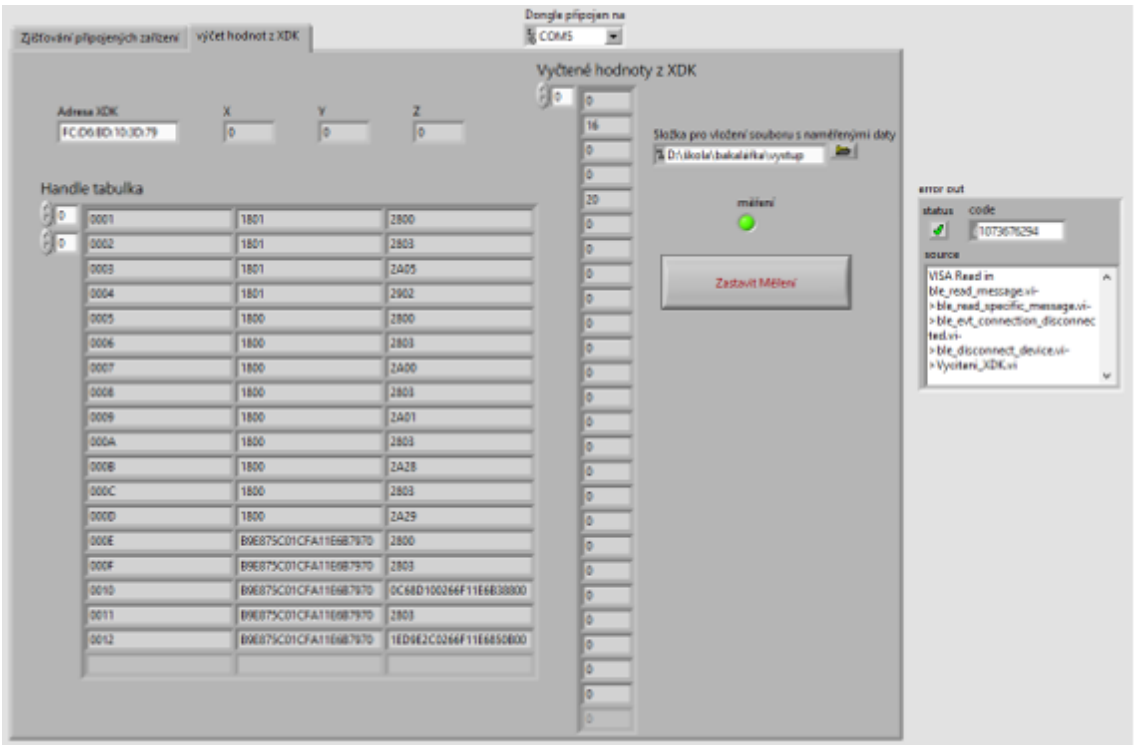
Aplikace umožňuje dvě volby spuštění. Jednou z nich je ‚Zjišťování připojených zařízení'. V této volbě je možno zjistit, zda požadované zařízení je aktivní a zda se na něj dá připojit, či nikoli. Také se zde dá nalézt adresa, kterou má zařízení XDK a kterou je potřeba nastavit při měření. Na Obr. 4-2 je ukázán vzhled aplikace na čelním panelu programu v LabVIEW a na Obr. 4-3 Kód programu pro volbu zjišťování připojených zařízení je ukázán kód, jakým tato volba funguje.



Obr. 4-2 Vzhled čelního panelu programu při volbě ‚Zjišťování připojených zařízení'

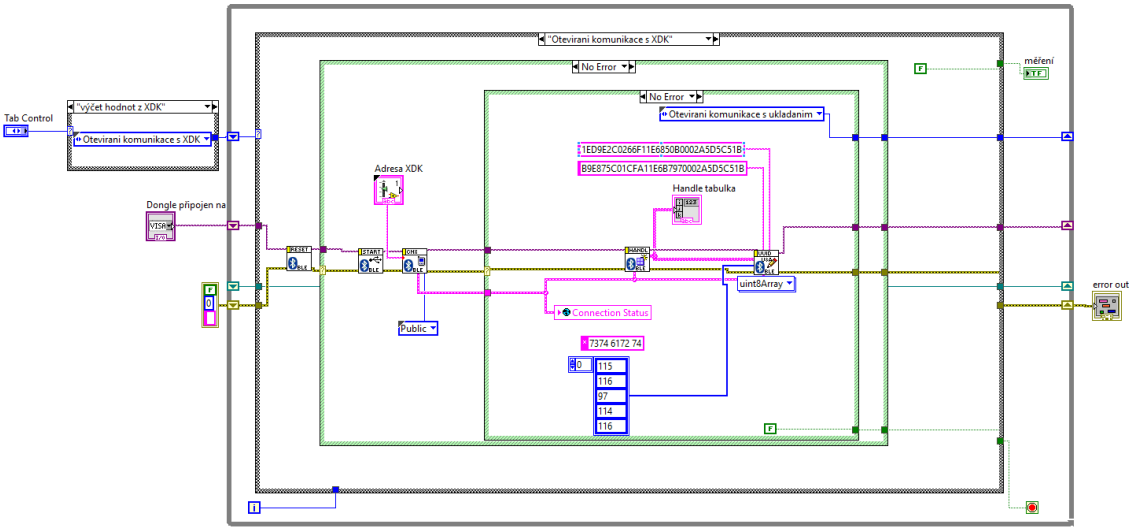


Obr. 4-3 Kód programu pro volbu zjišťování připojených zařízení

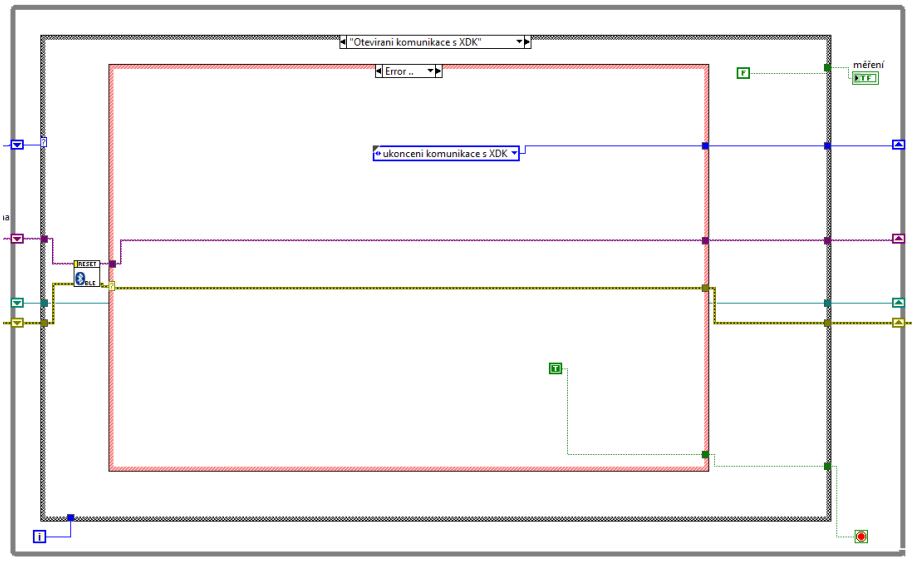


Obr. 4-4 Čelní panel – Volba měření

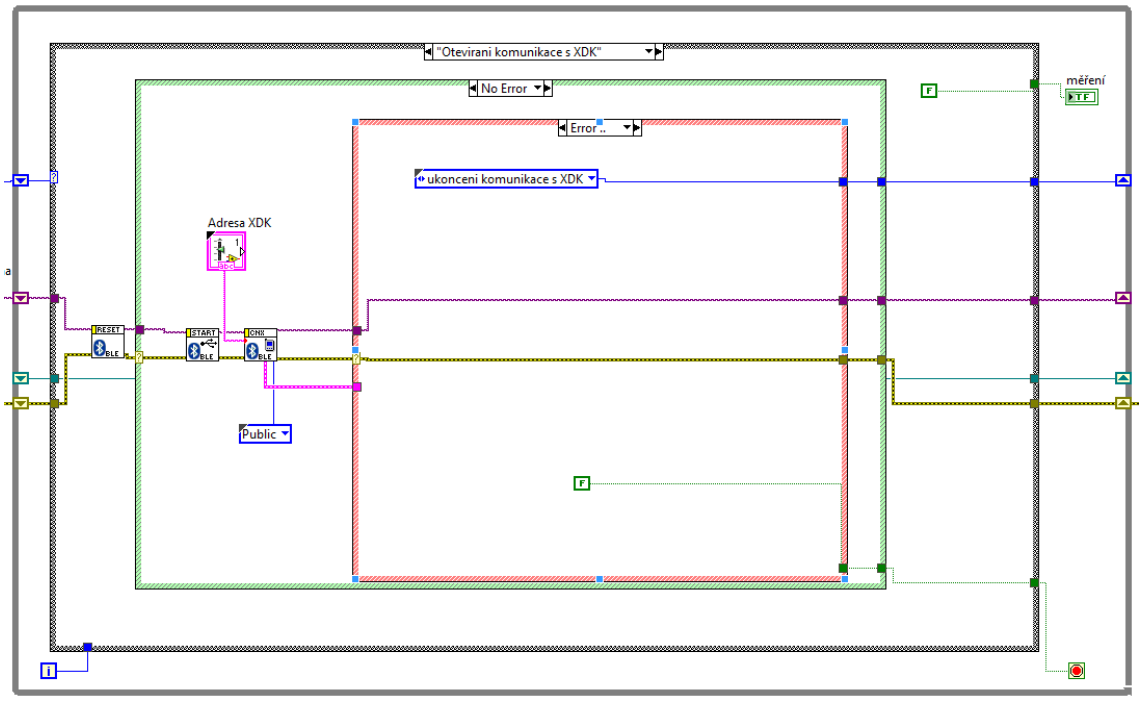
Volba měření. První částí je otevření komunikace s XDK, které je na Obr. 4-5. Zde se vyresetuje USB pro případ, že by nastalo dříve špatné ukončení komunikace s USB portem. Následně se USB port připojí a připojí se i k zařízení. Na konci části se poté vyšle do XDK kód pro začátek streamování dat. V případě, že se objeví chyba již při resetu USB, kód se automaticky ukončí. V případě, že se nepovede připojení k XDK, kód se hned ukončí také. Pokud vše proběhne, tak jak má, kód pokračuje do další části, kterou je otevření souboru pro ukládání.



Obr. 4-5 Blokový diagram – Otevření komunikace s XDK

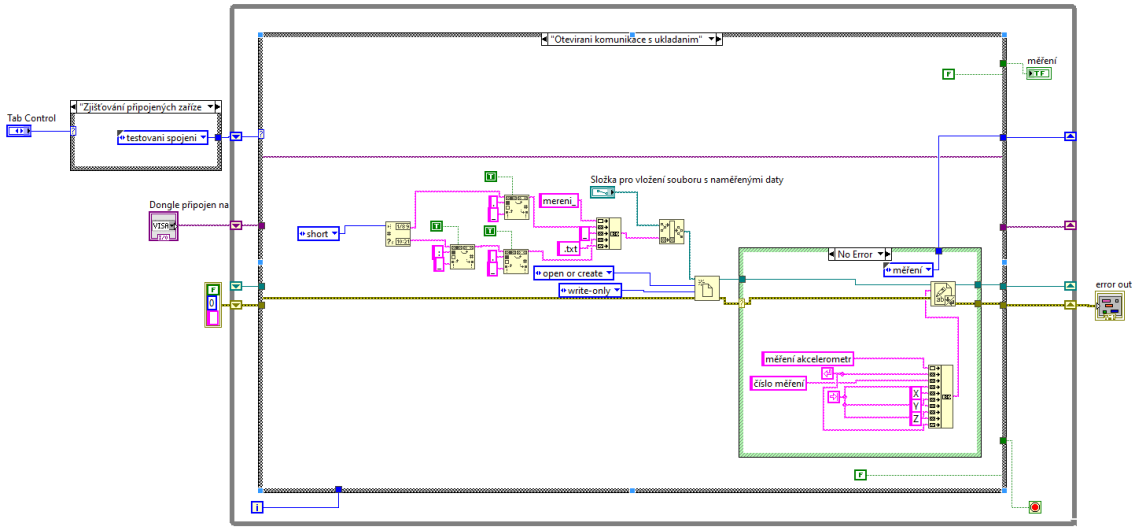


Obr. 4-6 Errorová case – Otevření komunikace s XDK

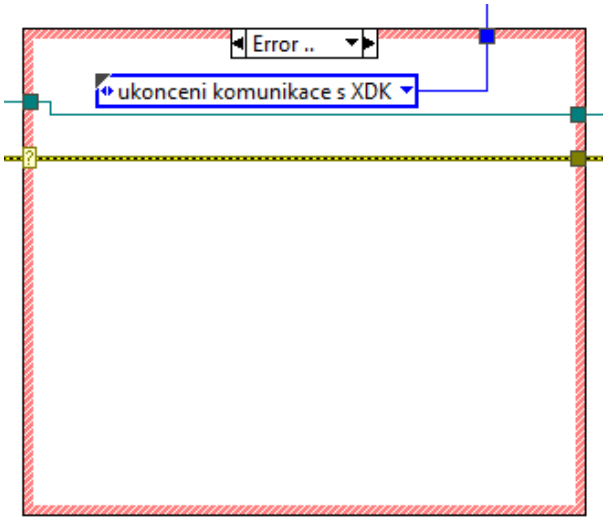


Obr. 4-7 Errorová case 2 - Otevření komunikace s XDK

Druhá část kódu je otevření/vytvoření souboru pro zápis naměřených hodnot, kdy v první části souboru se vygeneruje název txt souboru, do kterého se bude měření ukládat. Dále se tento soubor vytvoří. V případě, že se soubor povede vytvořit a otevřít, tak se do souboru zapíše hlavička a pokračuje se do sekce měření. V případě se nepovede otevření souboru pokračuje se na ukončení komunikace s XDK. Blokový diagram této části je vidět na Obr. 4-8.

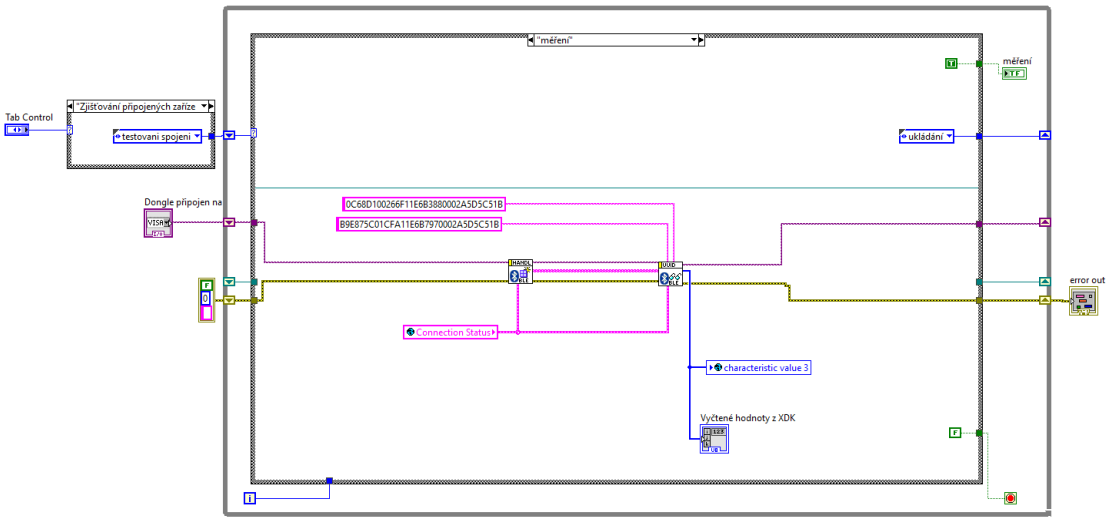


Obr. 4-8 Blokový diagram – Otevření souboru pro zápis naměřených hodnot



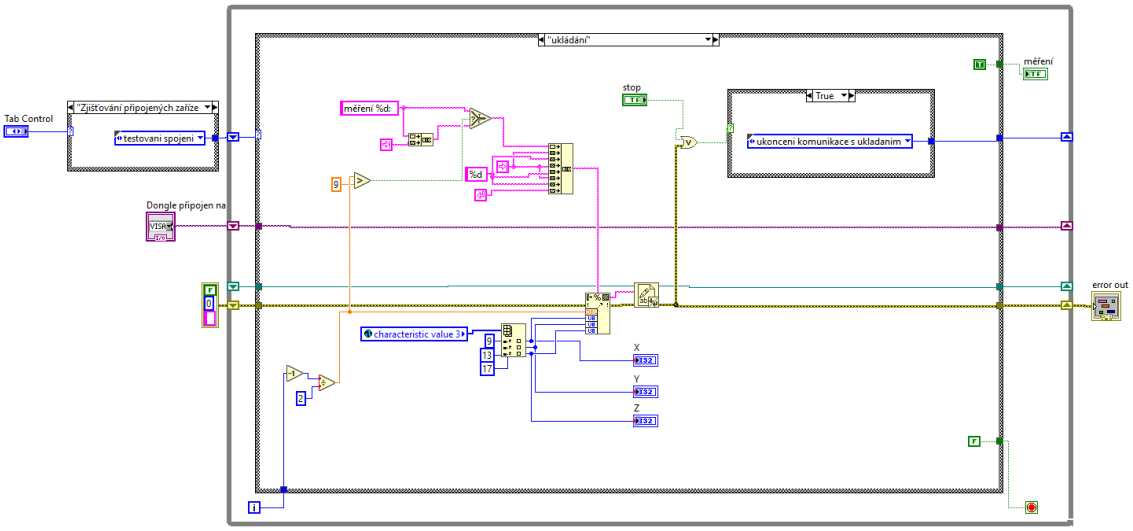
Obr. 4-9 Errorová case – Otevření souboru pro zápis

Třetí částí kódu je samotné měření, zde se vyčítají z XDK měřená data a ta se následně posílají na zpracování do sekce ukládání. Vzhled blokového diagramu této části je na Obr. 4-10.



Obr. 4-10 Blokový diagram – měření

Čtvrtou částí kódu je ukládání dat. Zde se vyčtená data upraví do správné podoby a následně se uloží do souboru, který byl otevřen v části kódu otevření/vytvoření souboru. Zde se také rozhoduje, zda program bude pokračovat či nikoli. Pokud je sepnuto tlačítko zastavit měření, program dále pokračuje do části uzavření souboru. Pokud tlačítko není sepnuto program se vrací do části měření. Blokový diagram této části je uveden na Obr. 4-11.



Obr. 4-11 Blokový diagram – ukládání

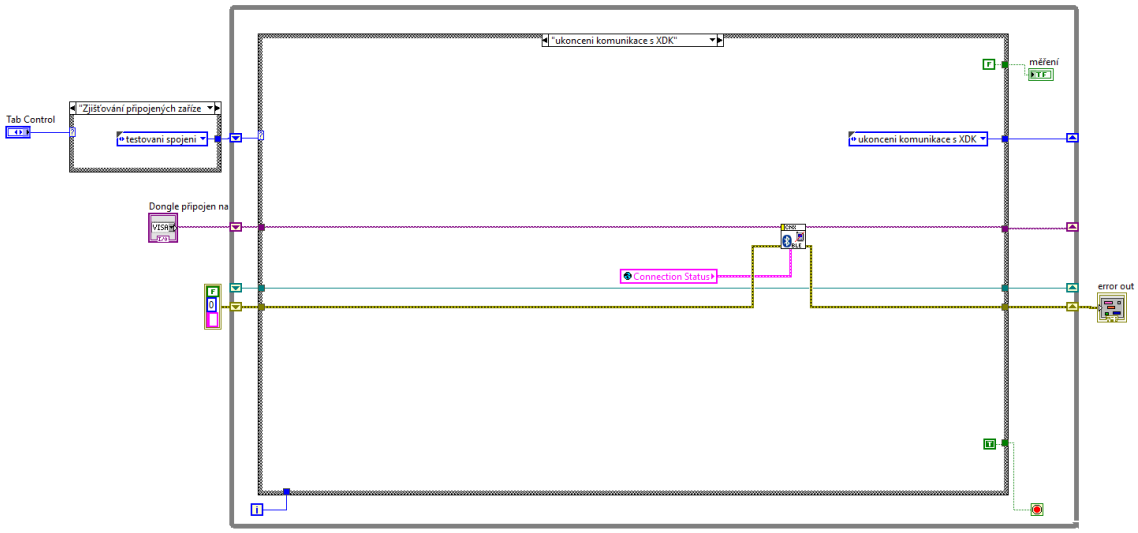
Pátou částí kódu je uzavření souboru, do kterého se zapisovaly naměřené hodnoty. Ta je uvedena na Obr. 1-2Obr. 4-12. Zde v kódu je ještě potřeba po zprovoznění přesně doladit konstanty, kde se nachází naměřené hodnoty.

Obr. 4-12 Blokový diagram – uzavření souboru pro ukládání dat měření

Poslední částí kódu je uzavření komunikace s XDK. Zde se prvně ukončí komunikace s XDK a následně se aplikace vypne. Kód je zobrazen na Obr. 4-13.

Součástí smyčky měřícího programu by mělo být ještě časování smyčky, aby nedocházelo k stoprocentnímu vytížení počítače pouze chodem tohoto programu.

Časování smyčky by mělo odpovídat požadované frekvenci vyčítání dat z XDK. V programu, který je přílohou této práce, je již tato chyba opravena.



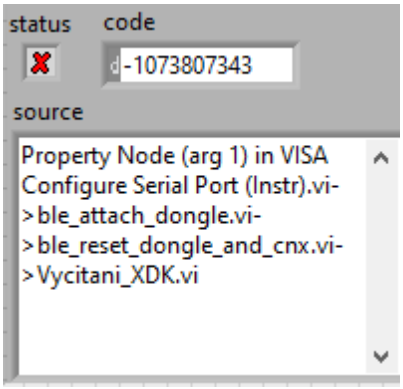
Obr. 4-13 Blokový diagram – ukončení komunikace s XDK

4.2 Možné chyby a jejich řešení

V této podkapitole si představíme možné chyby, které mohou nastat nebo nastaly při testování aplikace.

4.2.1 Připojení donglu a error -1073807343

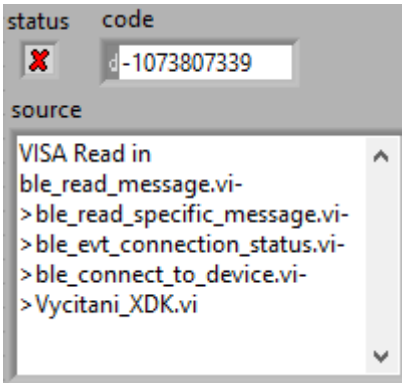
V případě, že není připojen dongle do správného USB, nebo není připojen vůbec, objeví se na výstupu kód chyby -1073807343 (na Obr. 4-14 Chyba kód -1073807343Obr. 4-14), tuto chybu lze odstranit připojením USB donglu.



Obr. 4-14 Chyba kód -1073807343

4.2.2 BLE v XDK není zapnuto a error -1073807339

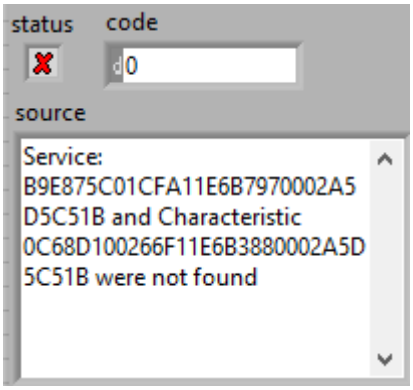
Tato chyba se objevuje v případě, že v XDK není zapnuto Bluetooth zařízení, toto může být způsobeno špatným programem v XDK, nebo přepnutím XDK do bootovacího režimu (k tématu přepnutí do bootovacího režimu je více uvedeno v kapitole 3.4.2). Tuto chybu lze vyřešit nahráním správného programu do XDK. Vzhled chyby je na Obr. 4-15.



Obr. 4-15 Chyba kód -1073807339

4.2.3 Špatné servisní nebo charakteristické UUID a error 0

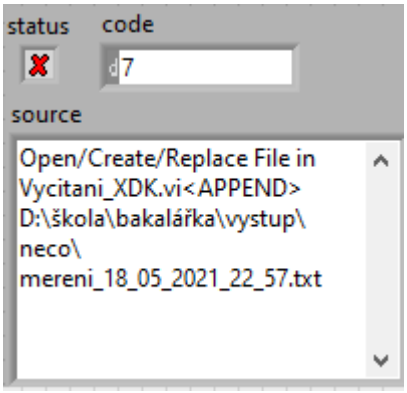
Tato chyba se objevuje v případě, že je v kódu špatně zadaný servisní nebo charakteristické UUID, případně obojí. Výstup chyby je uveden na Obr. 4-16. Postup úpravy UUID je v kapitole 3.4.3.



Obr. 4-16 Chyba kód 0 – servisní nebo charakteristické UUID

4.2.4 Neplatná složka pro ukládání hodnot a error 7

Tato chyba se objevuje v případě, že je v kódu na ukládání změřených hodnot uvedena složka, která neexistuje a program proto nemá kam vložit soubor s naměřenými hodnotami. Výstup chyby je uveden na Obr. 4-17.



Obr. 4-17 Chyba kód 7

Mimo výše uvedené chyby se mohou v kódu objevit i další, které se během vytváření kódu neprojevily.

4.3 Shrnutí funkčnosti aplikace

Vytvořená aplikace v LabVIEW zvládá zjistit, že zařízení XDK 110 je aktivní a je možno se k němu připojit. Dále je schopna nahrát handle tabulku XDK a následně do ní zapsat zprávu pro XDK. Aplikace má však problém s vyčítáním dat, kde by mohl být problém v protichůdných informacích v kódu využitého examplu v XDK-Workbench. Při prohledávání kódu jsem nalezl, že jsou v examplu uváděny jiné informace v komentářích a jiné v samotném kódu. Pro vyřešení problému s komunikací je možné, že by mohlo stačit pečlivé projití examplu zda se někde ještě nemění hodnota pro správný příkaz na spuštění streamování dat. Mimo tuto nefunkční část má aplikace funkční ukládání do souboru pro případné další zpracování.

5. ZÁVĚR

Hlavním cílem bakalářské práce bylo vytvořit návrh programu v LabVIEW pro stažení změřených dat z programovatelného zařízení XDK 110 při využití bezdrátového přenosu dat. Pro tuto bakalářskou práci bylo jako komunikační rozhraní zvoleno Bluetooth.

Vedlejším úkolem bylo vytvořit stručný popis zařízení Bosch XDK a návod na práci s ním. Tento návod se nachází v kapitole 1. V první části této kapitoly jsou popisovány jednotlivé snímače a druhá část popisuje program XDK-Workbench, který se používá k programování zařízení XDK. V této kapitole je také ukázán example, který byl v práci použit a probrána jeho struktura.

Kapitola 2 se zabývá komunikačním rozhraním Bluetooth a prací s Bluetooth v LabVIEW. Je zde popsána komunikace jak s klasickým Bluetooth, tak úprava pro komunikaci s Bluetooth Low Energy a také je zde popsáno, co Bluetooth Low Energy je. V práci jsou popsány dvě varianty řešení komunikace mezi BLE a LabVIEW, které použití BLE zatím přímo nepodporuje.

V kapitole 3 je uvedena původní představa aplikace, která se však z důvodu problémů s BLE změnila a aktuální verze aplikace je uvedena v kapitole 4. Mimo původní představy je v kapitole 3 uveden popis postupu řešení a také poznatky z řešení aplikace.

V kapitole 4 je popsána vypracovaná aplikace a jsou rozebrány chyby, které nastaly během řešení nebo se domnívám, že by se mohly během práce s aplikací vyskytnout.

Zadání práce nebylo splněno úplně. Nebylo dosaženo vyčtení hodnot z XDK, ale v práci je popsáno, kde ještě mohl nastat problém, který je potřeba ověřit a případně vyřešit, pro úspěšné vyčítání dat.

Pro následné pokračování s prací je možné dokončit vyčítání dat z XDK a zapracovat navržené scénáře pro vyčítání.

LITERATURA

[1] Cross Domain Development Kit. Bosch [online]. Gerlingen-Schillerhöhe: Bosch Connected Devices and Solutions, 2017 [cit. 2020-11-18]. Dostupné z: https://developer.bosch.com/documents/422133/482930/XDK_Node_110_combined_Datasheet.pdf/ace7ca7c-fd0e-9741-acf6-e9d5dc10957d

[2] Technical information. Bosch [online]. Gerlingen-Schillerhöhe: Bosch Connected Devices and Solutions, 2018 [cit. 2020-11-18]. Dostupné z: https://developer.bosch.com/web/xdk/technical-information

[3] BMG160. Alldatasheet [online]. [cit. 2020-12-30]. Dostupné z: https://pdf1.alldatasheet.com/datasheet-pdf/view/1131979/BOSCH/BMG160.html

[4] BMI160. Bosch [online]. Reutlinger: Bosch Sensortec, c2020 [cit. 2020-12-30]. Dostupné z: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bmi160-ds000.pdf

[5] BMA160. Bosch [online]. Reutlinger: Bosch Sensortec, c2020 [cit. 2020-12-30]. Dostupné z: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bma280-ds000.pdf

[6] Communicate with Bluetooth Low Energy (BLE) Devices in LabVIEW. NI Community [online]. Austin: NATIONAL INSTRUMENTS, c2020 [cit. 2020-12-30]. Dostupné z: https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z0000019LuHSAU&l=cs-CZ

[7] BLE Toolkit. NI Community [online]. Austin: NATIONAL INSTRUMENTS, c2020, 2016 [cit. 2020-12-30]. Dostupné z: https://forums.ni.com/t5/Community-Documents/LabVIEW-BLE-Bluetooth-Low-Energy-toolkit/ta-p/3538612?profile.language=en

[8] WHAT IS BLUETOOTH LOW ENERGY? ElaInnovation [online]. Montpellier: ELA Innovation, 2019 [cit. 2020-12-28]. Dostupné z: https://elainnovation.com/what-is-ble.html

[9] What is Bluetooth Low Energy (BLE) and how does it work? Centare [online]. Brookfield: Centare, c2019, 2019-03-07 [cit. 2020-12-30]. Dostupné z: https://www.centare.com/blog/what_is_bluetooth_low_energy/

[10] Anatomie programu Mita. Eclipse foundation [online]. Ottawa: Eclipse Foundation [cit. 2021-02-03]. Dostupné z: https://www.eclipse.org/mita/language/introduction/

[11] Bosch Connected Devices and Solutions prototypová deska XDK 110 Cross-Domain Development Kit XDK 110. Conrad Electronic [online]. Vinohradská 2828/151, 130 00, Praha 3 - Žižkov: Conrad Electronic Česká republika, c2020 [cit. 2021-5-4]. Dostupné z: https://www.conrad.cz/p/bosch-connected-devices-and-solutions-prototypova-deska-xdk-110-cross-domain-development-kit-xdk-110-1421124

[12] Bosch Connected Devices and Solutions presents XDK. Youtube [online]. Reutlingen: Bosch Connected Devices and Solutions, 2016 [cit. 2021-5-4]. Dostupné z: https://www.youtube.com/watch?v=FAlyjBO0-7g
[13] VLACH, Jaroslav, Josef HAVLÍČEK a Martin VLACH. Začínáme s LabVIEW. Praha: BEN – technická literatura, 2008. ISBN 978-80-7300-245-9.
[14] GATT. Adafruit [online]. adafruit, 2014 [cit. 2021-5-5]. Dostupné z: https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gatt
[15] Bluetoothapis.h header. Microsoft [online]. Redmond: Microsoft, c2021 [cit. 2021-5-9]. Dostupné z: https://docs.microsoft.com/en-us/windows/win32/api/bluetoothapis/
[16] Bluetooth vs WiFi Comparison For the IoT Solutions. Netguru [online]. Poznań: Netguru S.A., c2021 [cit. 2021-5-23]. Dostupné z: https://www.netguru.com/codestories/bluetooth-vs-wifi-comparison-for-the-iot-solutions
[17] AKU340. Mouser Electronic [online]. Brno: Mouser Electronics, c2021 [cit. 2021-5-23]. Dostupné z: https://www.mouser.com/datasheet/2/720/DS26-1.04%20AKU340%20Datasheet-770074.pdf
[18] GATT (Services and Characteristics). O'Reilly [online]. Sebastopol: O'Reilly Media, c2021 [cit. 2021-5-23]. Dostupné z: https://www.oreilly.com/library/view/getting-started-with/9781491900550/ch04.html#gatt_svc_disc

SEZNAM SYMBOLŮ A ZKRATEK

Zkratky:

ATT	Attribute protocol
BLE	Bluetooth Low Energy
DC	Direct Current, stejnosměrný proud
DEC	dekadické číslo
GATT	Generic Attribute
GPIB	General Purpose Interface Bus
HEX	hexadecimální číslo
IoT	Internet of Thinks
OS	Operační systém
txt	koncovka textového souboru
USB	Universal Serial Bus (Univerzální sériová sběrnice)
UUID	universally unique identifier
VI	Virtual Instument, přípona souborů LabVIEW
WEP	Wired Equivalent Privacy, zastaralý standard zabezpečení bezdrátové sítě
WPA	Wi-Fi Protected Access, zabezpečení bezdrátových sítí

SEZNAM PŘÍLOH